Use of Learned Dictionaries in Tomographic Reconstruction

Vincent Etter[a], Ivana Jovanović[b,c] and Martin Vetterli[d]

[a]Laboratory for Communications and Applications, Ecole Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland

[b]Medical Image Processing Lab, Ecole Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland

[c]Medical Image Processing Lab, University of Geneva, CH-1211 Geneva, Switzerland

[d]Audiovisual Communications Laboratory, Ecole Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland

ABSTRACT

We study the use and impact of a dictionary in a tomographic reconstruction setup. First, we build two different dictionaries: one using a set of bases functions (Discrete Cosine Transform), and the other that is learned using patches extracted from training images, similar to the image that we would like to reconstruct. We use K-SVD as the learning algorithm. These dictionaries being local, we convert them to global dictionaries, ready to be applied on whole images, by generating all possible shifts of each atom across the image. During the reconstruction, we minimize the reconstruction error by performing a gradient descent on the image representation in the dictionary space. Our experiments show promising results, allowing to eliminate standard artifacts in the tomographic reconstruction, and to reduce the number of measurements required for the inversion. However, the quality of the results depends on the convergence of the learning process, and on the parameters of the dictionaries (number of atoms, convergence criterion, atom size, etc.). The exact influence of each of these remains to be studied.

Keywords: dictionary learning, inverse problems, ultrasound tomography

1. INTRODUCTION

In this article, we study the benefits of using a dictionary in the context of inverse problems, more specifically in the tomography setup. We chose an emerging application of ultrasound tomography in breast screening to practically demonstrate possible improvements in the image quality when specific learned dictionaries are used in the reconstruction process. Assuming that the image to be reconstructed has a sparse representation in the space of a learned dictionary the reconstruction process can be improved in two ways. First, the image can be reconstructed with higher resolution because a smaller data set is needed compared to the requirements for the same resolution and without the learned dictionary. Second, if we have a noisy data set the image will have better quality when reconstructed with the learned dictionary than without. The idea behind these benefits is that learned dictionaries will allow for more efficient representation of an image and therefore the image can be represented in a lower dimensional space. Thus, because of this dimensionality reduction less measurements will be needed to recover this image.

Throughout this paper we cover the application of learned dictionaries in ultrasound tomography by going from theoretical questions on how to learn dictionaries to the very practical issues of using them in this concrete application. The paper is organized as follows. In Section 2, we first introduce the notion of sparse coding, its different variations and the main approaches to solve it. In Section 3, we present the two families of dictionaries: complete bases and learned dictionaries. We describe one example of complete basis composed of Discrete Cosine

Further author information: (Send correspondence to V. Etter)
V. Etter - email: vincent.etter@epfl.ch
I. Jovanović - email: ivana.jovanovic@epfl.ch
M. Vetterli - email: martin.vetterli@epfl.ch, phone: +41 21 693 5698

Wavelets and Sparsity XIV, edited by Manos Papadakis, Dimitri Van De Ville, Vivek K. Goyal, Proc. of SPIE Vol. 8138, 81381C · © 2011 SPIE · CCC code: 0277-786X/11/$18
doi: 10.1117/12.894776

Transform basis functions, and then list the main techniques for dictionary learning. We focus particularly on K-SVD, as it shows good results in many applications, and has a complexity-wise efficient implementation. We present our tomography setup and the application of these dictionaries in Section 4. We first describe how learned dictionaries are trained on patches extracted from medical images obtained with other imaging modalities. We then explain how the local atoms of these dictionaries are "globalized" to be applied on whole images. Finally, we explain how the original reconstruction problem should be adapted such that the learned dictionary can be used. Reconstruction results are presented in Section 5, in which we compare the different dictionaries, and also explore the influence of the dictionary parameters (patch size, number of atoms) on the reconstruction quality. Finally, we conclude this article in Section 6 with some discussion and perspectives of future work.

2. SPARSE REPRESENTATION OF SIGNALS

Let $\boldsymbol{y} \in \mathbb{R}^n$ be a signal of any kind (image, sound, MRI measurement, etc). Using an overcomplete dictionary matrix $\boldsymbol{D} \in \mathbb{R}^{n \times K}$, with $K > n$, that contains K columns $\{\boldsymbol{d}_j\}_{j=1}^K$ (called atoms), $\boldsymbol{y}$ can be represented as a sparse linear combination of these atoms, such that this representation is either exact :

$$\boldsymbol{y} = \boldsymbol{D}\boldsymbol{x}, \tag{1}$$

or approximate :

$$\boldsymbol{y} \approx \boldsymbol{D}\boldsymbol{x} \text{ with } \|\boldsymbol{y} - \boldsymbol{D}\boldsymbol{x}\|_p \leq \epsilon. \tag{2}$$

$\boldsymbol{D}$ being overcomplete, an infinite number of solutions are available for the representation problem. The solution with the fewest number of nonzero coefficients is certainly an appealing representation. This sparsest representation is the solution of either :

$$\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_0 \text{ with constraint } \boldsymbol{y} = \boldsymbol{D}\boldsymbol{x}, \tag{3}$$

or

$$\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_0 \text{ with constraint } \|\boldsymbol{y} - \boldsymbol{D}\boldsymbol{x}\|_p \leq \epsilon, \tag{4}$$

with $\|\cdot\|_0$ being the L^0 norm, counting the number of nonzero entries of a vector.

2.1 Sparse coding

Sparse coding is the process of computing the representation $\boldsymbol{x}$ based on the signal $\boldsymbol{y}$ and the dictionary $\boldsymbol{D}$. This process, commonly called "atom decomposition", requires solving Eq. (1) or Eq. (2), usually using a "pursuit algorithm" that finds an approximate solution.

Indeed, exact determination of the sparsest representation proves to be NP-hard. Thus, approximations are considered instead.

2.1.1 Greedy approach

Matching Pursuit (MP) and Orthogonal Matching Pursuit (OMP)[1] are the two simplest algorithms commonly used to solve the atom decomposition problem. They basically select the dictionary atoms sequentially, and involve computing the inner product between the signal and dictionary columns. By setting the stopping rule of the algorithm, one can either find a solution to Eq. (1) or Eq. (2).

If the approximation delivered has k_0 nonzero coefficients, OMP has a complexity of $O(k_0 n K)$.

2.1.2 Convexification approach

A second well-known approach consists of replacing the L^0 norm with a L^1 norm (Basis Pursuit (BP)), or with a $L^p(p \leq 1)$ norm (Focal Underdertermined System Solver (FOCUSS)). Lagrange multipliers are used to convert the constraint into a penalty term, and an iterative method is used to solved the minimisation problem.

These algorithms have been shown to recover well the signal, given that the sought solution $\boldsymbol{x}$ is sparse enough.

3. CHOICE OF THE DICTIONARY

To solve the minimization problems in Eqs. (3) and (4), one should first decide how to build the dictionary $\boldsymbol{D}$. The choice of the dictionary will greatly influence the sparsity of $\boldsymbol{x}$ and the quality of the reconstructed signal.

There are two different approaches for choosing this dictionary : the *complete bases* approach, in which a prespecified set of functions are used, or the *learning-based* approach, in which a set of function is designed to fit a given set of signal examples.

3.1 Complete bases

Choosing a prespecified transform matrix is appealing because it is simpler. Also, in many case it leads to simple and fast algorithms for the evaluation of the sparse representation. This is indeed the case for overcomplete wavelets, curvelets, short-time Fourier transforms, and more.

The success of such dictionaries in applications depends on how suitable they are to sparsely describe the signals in question. While used for many years, this analytic approach has been frequently outperformed by dictionaries based on learning.

3.1.1 Discrete Cosine Transform

To compare with learned dictionaries, we used Discrete Cosine Transform (DCT)[2] as a complete basis to build a dictionary. In one dimension, DCT defines a set of N basis functions as follows:

$$f_k(x) = cos\left[\frac{\pi}{N}(x+\frac{1}{2})k\right], k = 0, \ldots, N-1.$$

These bases can be easily extended to 2 dimensions, by taking two sets of 1D basis functions and multiplying them two by two. Figure 1 shows an example of such a dictionary, with 81 atoms of 8×8 pixels.

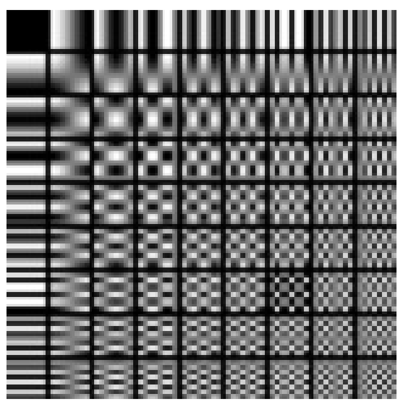


Figure 1. Example of an overcomplete set of 81 2D DCT basis functions, for patches of size 8×8.

3.2 Learning-based approach

In this approach, that has recently led to state-of-the-art results in many applications, the dictionary is built such that it leads to a sparse representation of the training signals. There are many ways to build this dictionary. The main ones are described in the following.

3.2.1 Online learning

Most algorithms take the training set as a whole when using it for learning the dictionary. In real-world applications, for instance video processing, the dataset may contain millions of samples, thus rendering impossible the use of a batch algorithm.

Online algorithms using a stochastic approach have been proposed by Bottou.[3] They randomly sample the training set and use at each iteration only one sample to update the dictionary. This approach has been shown to be significantly faster than batch algorithms, and to achieve similar results.

An online algorithm for dictionary learning and sparse coding is proposed by Mairal.[4]

3.2.2 Component analysis

Principal Component Analysis (PCA) is an orthogonal linear transformation that projects the data into a new coordinate system such that its first component in this new representation (called the first principal component) explains the greatest variance of the data, the second coordinate the second greatest variance, etc. It can be computed using the singular value decomposition of the data matrix, and thus may be computationally expensive. The main drawback of PCA is that because it produces a complete set of orthogonal bases, it sometimes cannot model properly the data.

There exist different variations of the PCA algorithm:

- Sparse Principal Component Analysis (SPCA),[5] in which the output is constrained to be sparse.
- Independent Component Analysis (ICA),[6] that allows the learning of non-orthogonal bases, thus allowing to model properly data having a non-Gaussian distribution.

The main limitation of this family of algorithms is that it produces complete bases, that may sometimes not be sufficient to allow a good representation of the data, as opposed to an overcomplete basis.

3.2.3 K-Means

K-Means is a clustering algorithm, in which a set of descriptive vectors $\{\boldsymbol{d}_k\}_{k=1}^{K}$ is learned, and each sample is represented by the closest of those vectors (usually in the L^2 distance measure). This can be seen as an *extreme sparse representation*, in which only one of the coefficient is nonzero and is equal to one.

The algorithm is performed by iteratively applying two steps :

1. assign each sample $\boldsymbol{x}_i$ to the closest descriptive vector $\boldsymbol{d}_k$
2. updates the vectors $\boldsymbol{d}_k$ such that they better represent their assigned $\boldsymbol{x}_i$ (usually by using their mean)

This two-steps approach inspired many other algorithms, that usually only differ in the way they assign the $\boldsymbol{x}_i$ or update the $\boldsymbol{d}_k$.

3.2.4 Maximum Likelihood

This method uses probabilistic reasoning for reconstructing the dictionary $\boldsymbol{D}$. It assumes that for every example $\boldsymbol{y}$:

$$\boldsymbol{y} = \boldsymbol{D}\boldsymbol{x} + \boldsymbol{v},$$

with $\boldsymbol{x}$ a sparse representation and $\boldsymbol{v}$ Gaussian white noise with variance σ^2.

Given the examples $\boldsymbol{Y} = \{\boldsymbol{y}_i\}_{i=1}^{N}$, this method want to maximize the likelihood $P(\boldsymbol{Y}|\boldsymbol{D})$ by finding the best $\boldsymbol{D}$, with constraint that the columns of the representation $\boldsymbol{X}$ are sparse.

The problem can be expressed as follows :

$$\begin{aligned} \boldsymbol{D} &= \arg\max_{\boldsymbol{D}} \sum_{i=1}^{N} \max_{\boldsymbol{x}_i} \{P(\boldsymbol{y}_i, \boldsymbol{x}_i|\boldsymbol{D})\} && (5) \\ &= \arg\min_{\boldsymbol{D}} \sum_{i=1}^{N} \min_{\boldsymbol{x}_i} \left\{ \|\boldsymbol{D}\boldsymbol{x}_i - \boldsymbol{y}_i\|_2^2 + \lambda \|\boldsymbol{x}_i\|_1 \right\}. && (6) \end{aligned}$$

This minimisation problem can be approximated using an iterative algorithm, where first the $\boldsymbol{x}_i$ are calculated using a gradient descent, and then the dictionary $\boldsymbol{D}$ is updated to reflect better the new $\boldsymbol{x}_i$:

$$\boldsymbol{D}^{(n+1)} = \boldsymbol{D}^{(n)} - \eta \sum_{i=1}^{N} \left(\boldsymbol{D}^{(n)}\boldsymbol{x}_i - \boldsymbol{y}_i \right) \boldsymbol{x}_i^T.$$

3.2.5 Method of Optimal Directions (MOD)

This algorithm follows the K-Means philosophy, in which first a coding stage is performed using any pursuit algorithm (generally OMP or FOCUSS), and then the dictionary is updated to fit better the obtained representation. The main contribution of MOD is its simpler way of updating the dictionary.

After the coding stage, we define for each sample the error $\boldsymbol{e}_i = \boldsymbol{y}_i - \boldsymbol{D}\boldsymbol{x}_i$. The overall representation mean square error is thus given by :

$$\|\boldsymbol{E}\|_F^2 = \|[\boldsymbol{e}_1, \boldsymbol{e}_2, \cdots, \boldsymbol{e}_n]\|_F^2 = \|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F^2 ,$$

where the notation $\|\boldsymbol{A}\|_F^2$ stands for the Frobenius norm, defined as $\|\boldsymbol{A}\|_F^2 = \sqrt{\sum_{ij} \boldsymbol{A}_{ij}^2}$.

Assuming $\boldsymbol{X}$ is fixed, we can update $\boldsymbol{D}$ in order to minimise this error. By deriving the above equation with respect to $\boldsymbol{D}$ and equaling it to zero, we obtain the relation $(\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X})\boldsymbol{X}^T = 0$. The resulting update rule for $\boldsymbol{D}$ is thus :

$$\boldsymbol{D}^{(n+1)} = \boldsymbol{Y}\boldsymbol{X}^{(n)^T} \cdot \left(\boldsymbol{X}^{(n)}\boldsymbol{X}^{(n)^T}\right)^{-1} .$$

3.2.6 Maximum A-Posteriori

Similarly to the maximum likelihood method, this approach uses a probabilistic view of the problem. However, instead of maximising the likelihood of the output given the dictionary $P(\boldsymbol{Y}|\boldsymbol{D})$, the posterior $P(\boldsymbol{D}|\boldsymbol{Y})$ is used. Using Bayes' rule, we have $P(\boldsymbol{D}|\boldsymbol{Y}) \propto P(\boldsymbol{Y}|\boldsymbol{D})P(\boldsymbol{D})$. Thus, we can use the likelihood expression as before, and add a prior on the dictionary.

Compared to MOD, this kind of algorithm provides slower training methods.

3.2.7 Union of Orthonormal Bases

This method, introduced by Elad[7] and generalized by Gribonval,[8] exploits the fact that any signal $\boldsymbol{y} \in \mathbb{R}^n$ has a unique representation in different orthonormal bases $\boldsymbol{D}^{(i)} \in \mathbb{R}^{n\times n}$. Let $\boldsymbol{D}^{(1)}, \boldsymbol{D}^{(2)}, \cdots, \boldsymbol{D}^{(L)}$ be L of these bases. We can write :

$$\boldsymbol{y} = \sum_{i=1}^{n} \boldsymbol{D}_{(i)}^{(1)}\boldsymbol{x}_i^{(1)} = \boldsymbol{D}^{(1)}\boldsymbol{x}^{(1)} = \boldsymbol{D}^{(2)}\boldsymbol{x}^{(2)} = \cdots = \boldsymbol{D}^{(L)}\boldsymbol{x}^{(L)},$$

where $\boldsymbol{x}^{(i)} \in \mathbb{R}^n$ is the representation of $\boldsymbol{y}$ in the base $\boldsymbol{D}^{(i)}$. By taking the union of these L bases as an overcomplete dictionary

$$\boldsymbol{D} = \left[\boldsymbol{D}^{(1)}|\boldsymbol{D}^{(2)}|\cdots|\boldsymbol{D}^{(L)}\right] \in \mathbb{R}^{n\times nL}$$

and setting

$$\boldsymbol{y}^* = \left[\boldsymbol{y}^T|\boldsymbol{y}^T|\cdots|\boldsymbol{y}^T\right]^T \in \mathbb{R}^{nL}$$

$$\boldsymbol{x} = \left[\boldsymbol{x}^{(1)^T}|\boldsymbol{x}^{(2)^T}|\cdots|\boldsymbol{x}^{(L)^T}\right]^T \in \mathbb{R}^{nL},$$

we obtain the following system : $\boldsymbol{y}^* = \boldsymbol{D}\boldsymbol{x}$.

It has been shown by Bruckstein[9] that by carefully choosing the bases $\boldsymbol{D}^{(i)}$ (such that they have high mutual coherence), we can have a sparse representation $\boldsymbol{x}$ of $\boldsymbol{y}^*$ in the overcomplete dictionary $\boldsymbol{D}$. Moreover, this structure allows for a more effective dictionary update, leading to a faster training.

3.2.8 K-SVD

K-SVD is a generalisation of the K-Means algorithm, and thus follows the same two-steps iterating process. In the first step, called the *Sparse Coding stage*, any pursuit algorithm can be used to compute the representation $\boldsymbol{x}$, allowing it to have at most T_0 non-zero coefficients :

$$\min_{\boldsymbol{x}} \left\{\|\boldsymbol{y}_i - \boldsymbol{D}\boldsymbol{x}\|_2^2\right\} \text{ subject to } \|\boldsymbol{x}\|_0 \leq T_0.$$

The main difference with K-Means resides in the second step, called the *Codebook Update Stage* : while K-Means freezes the x to update the dictionary D, K-SVD changes the columns of D sequentially, and allows the relevant coefficients of x to be updated as well.

Update procedure Here is the detailed update procedure of K-SVD, done for each atom $\boldsymbol{d}_k, k = 1, ..., K$ in the dictionary $\boldsymbol{D}^{(j-1)}$, obtained from the previous iteration of the algorithm :

1. Define the set of training samples that use this atom $\boldsymbol{d}_k$:
$$w_k = \{i | 1 < i < N, x_i(k) \neq 0\}$$
(where x_i is the ith training sample and $x_i(k)$ is the kth coefficient of this sample).
2. Compute the residual error $\boldsymbol{e}_k^{(i)}$, corresponding to the atom k, of each sample $\boldsymbol{x}_i, i \in w_k$:
$$\boldsymbol{e}_k^{(i)} = y_i - \sum_{j \neq k} d_j x_i(j)$$
$$\boldsymbol{E}_k = \left[\boldsymbol{e}_k^{(i_1)} \boldsymbol{e}_k^{(i_2)} \cdots \boldsymbol{e}_k^{(i_m)}\right], i_j \in w_k$$
3. Apply SVD decomposition to $\boldsymbol{E}_k$:
$$\boldsymbol{E}_k = \boldsymbol{U} \Delta \boldsymbol{V}^T$$
4. Choose the updated atom $\boldsymbol{d}_k$ to be the first column of $\boldsymbol{U}$: $\boldsymbol{d}_k = \boldsymbol{U}_1$
5. Update the kth coefficient of each sample $\boldsymbol{x}_i, i \in w_k$ to be the corresponding coefficient in the first column of $\Delta(1,1)\boldsymbol{V}$.

Implementation The implementation of both approximate K-SVD and Batch-OMP presented by Rubinstein[10] were used.

4. APPLICATION TO INVERSE PROBLEMS

4.1 Tomography setup

The inverse problem that we focused on is acoustic or ultrasound tomography. In this setup, we use emitters and receivers, placed on a circle around an object of interest. Each emitter sends sequentially a sound, and all receivers record for each emitter the time of flight of the sound, *i.e.* how long it took to cross between the two. If we have n emitters/receivers, we measure n^2 times of flight. The goal is then to recover the sound speeds inside the area of interest.

To do so, we divide the area between the emitters/receivers in a grid of size $m \times m$, and consider that the sound speed is constant inside each cell. Then, we build a measurement matrix $\boldsymbol{M}$ of size $n^2 \times m^2$. This matrix describes, for each "ray" going from one emitter to one receiver, the length of the segment that crosses each cell. Figure 2 shows a simplified example of such a setup.

The n^2 times of flight are recorded in the vector $\boldsymbol{t}_{of}$, which gives us the following equation:
$$\boldsymbol{M}\boldsymbol{x} = \boldsymbol{t}_{of},$$
where $\boldsymbol{x}$ is a vector of size m^2 containing the inverse sound speed in each cell of the grid, *i.e.* $\boldsymbol{x} = \left(\frac{1}{c_1} \frac{1}{c_2} \cdots \frac{1}{c_{m^2}}\right)$.

4.2 Solving the inverse problem

The current approach for solving this problem is an iterative method that uses gradient descent to find the best solution $\boldsymbol{x}$ to this equation, with fixed $\boldsymbol{M}$ and measured $\boldsymbol{t}_{of}$. At each step, we compute the cost function $f_c(\boldsymbol{x})$, which is usually the L^2 norm of the error:
$$f_c(\boldsymbol{x}) = \|\boldsymbol{M}\boldsymbol{x} - \boldsymbol{t}_{of}\|^2$$
To compute the update of the sound speeds for the next step, we simply take the derivative of the cost function:
$$\frac{\delta f_c(\boldsymbol{x})}{\delta \boldsymbol{x}} = 2(\boldsymbol{M}\boldsymbol{x} - \boldsymbol{t}_{of})^T \boldsymbol{M}$$
and update $\boldsymbol{x}$ by incrementing it by this gradient (or a scaled version of the gradient).

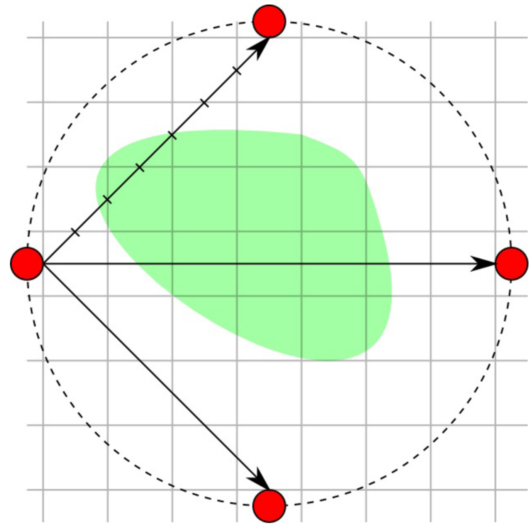


Figure 2. Example of the tomography setup. Each emitter/receiver (red circles) emits a sound, that the other receivers record. The path of the sound can be seen as a ray. We measure the length of the portions of one ray that cross each cell of the grid to build the matrix $\boldsymbol{M}$.

4.3 Usage of the dictionary

We would now like to introduce a dictionary in this tomography setup. To do so, we represent the sound speeds $\boldsymbol{c}$ as a linear combination of the columns of some dictionary $\boldsymbol{\psi}$:

$$\boldsymbol{c} = \boldsymbol{\psi}\boldsymbol{\theta}.$$

The dictionary $\boldsymbol{\psi}$ either comes from an overcomplete basis, or is learned using some problem-specific data, as explained above. The unknowns that we are now interested in are the coefficients in the dictionary space $\boldsymbol{\theta}$.

As we have seen in the previous section, dictionary atoms correspond to small patches of images. While such dictionaries can be used directly for problems like image denoising, it is not the case for the tomography setup. Indeed, we need to have a full dictionary to represent the sound speeds in the whole image at once, *i.e.* $\boldsymbol{c} = \boldsymbol{\psi}\boldsymbol{\theta}$.

Thus, from our initial learned dictionary $\boldsymbol{D}$, we build a new dictionary $\boldsymbol{\psi}$, that consists of all the shifted versions of each atom of $\boldsymbol{D}$. To do so, for each local atom of size $b \times b$, we generate $(n - b + 1)^2$ global atoms of size $n \times n$, by shifting it across the image at all possible positions, and filling the rest of the atom with zeros. This process is illustrated in figure 3.

Now, we can replace the sound speeds vector $\boldsymbol{c}$ with $\boldsymbol{\psi}\boldsymbol{\theta}$ in the equations above, which gives:

$$\boldsymbol{M}\boldsymbol{\psi}^\star\boldsymbol{\theta}^\star = \boldsymbol{t}_{of},$$

where $\boldsymbol{\psi}^\star$ and $\boldsymbol{\theta}^\star$ are respectively the elementwise inverse of $\boldsymbol{\psi}$ and $\boldsymbol{\theta}$, i.e. $\boldsymbol{\psi}^\star_{ij} = \frac{1}{\boldsymbol{\psi}_{ij}}$ and $\boldsymbol{\theta}^\star_i = \frac{1}{\boldsymbol{\theta}_i}$. The cost function is changed to

$$f_c(\boldsymbol{\theta}) = \|\boldsymbol{M}\boldsymbol{\psi}^\star\boldsymbol{\theta}^\star - \boldsymbol{t}_{of}\|^2$$

and the gradient now becomes

$$\frac{\delta f_c(\boldsymbol{\theta})}{\delta\boldsymbol{\theta}} = 2(\boldsymbol{M}\boldsymbol{\psi}^\star\boldsymbol{\theta}^\star - \boldsymbol{t}_{of})^T\boldsymbol{M}(\boldsymbol{\psi}^\star\boldsymbol{\theta}^\star).^2\boldsymbol{\psi},$$

where $(\cdot).^2$ means squared elementwise. $\boldsymbol{\theta}^\star$ is updated as before, by incrementing it by the gradient above (or a scaled version of it).

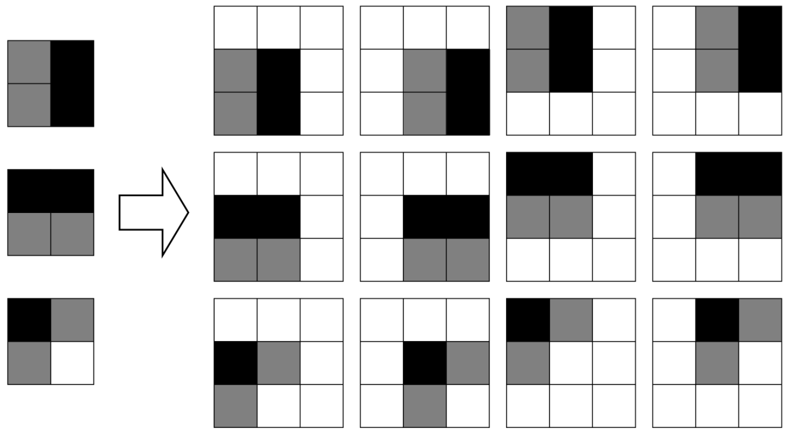


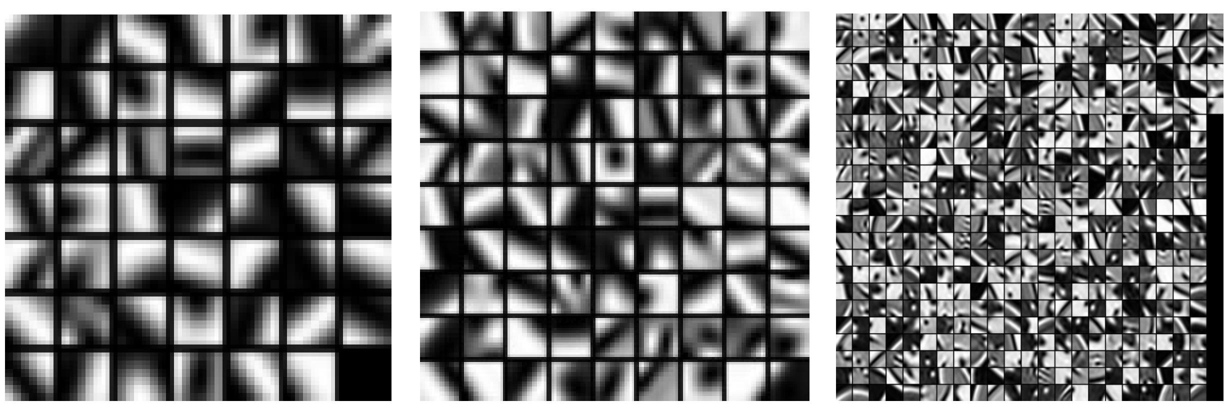
Figure 3. Expansion of the learned dictionary $\boldsymbol{D}$ (left), that contains atoms of size $b \times b$, to a global dictionary $\boldsymbol{\psi}$ (right) that contains atoms of size $n \times n$, and is generated by shifting each atom of $\boldsymbol{D}$ to all possible positions in the image.

5. RESULTS

5.1 Dictionary learning

To learn dictionaries for our experiments, we used a dataset of medical images, similar to the ones we would have to reconstruct. For different combinations of patch size and number of atoms, we extracted 100'00 patches from these images, and applied K-SVD on them to learn a dictionary.

Figure 4 shows examples of such learned dictionaries, for different numbers of atoms, and different patch sizes.



(a) 48 atoms, 6×6 (b) 81 atoms, 8×8 (c) 512 atoms, 16×16

Figure 4. Examples of dictionaries learned using K-SVD on 100'000 patches extracted from a dataset of medical images. Different setup are illustrated, with the patch sizes and the number of atoms varying.

5.2 Reconstruction results

To evaluate the impact of the dictionaries, we picked one 144×144 image similar to the training images. We then ran our tomography software to simulate measurements from 256 emitters/receivers placed on a circle around the image, and finally ran the inversion algorithm on these measurements, to try and reconstruct the original image. We compared three experiments:

1. inversion without a dictionary (INV)
2. inversion with a DCT dictionary (INV-DCT)

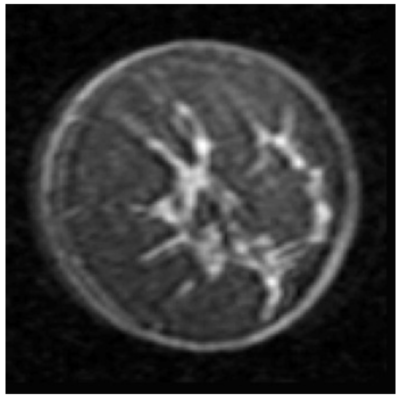
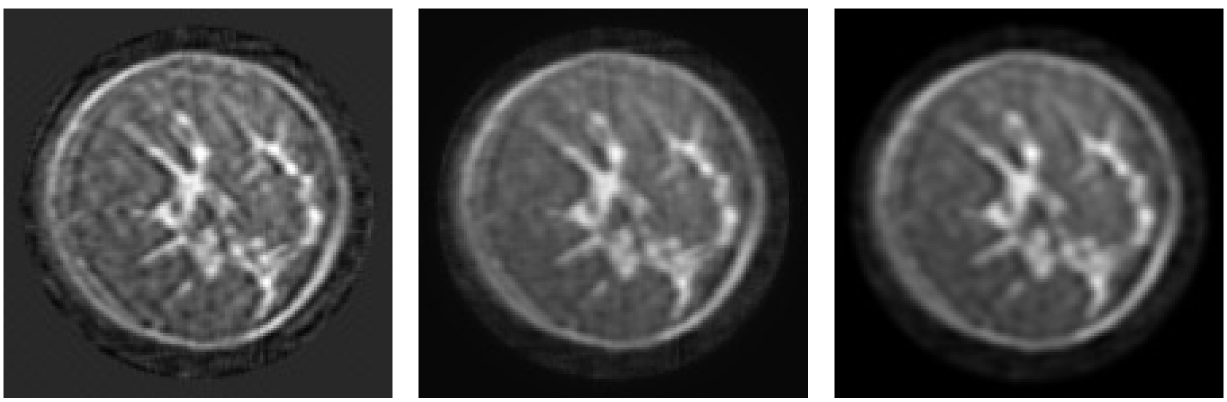


Figure 5. Original image used for tomography experiments.

3. inversion with a dictionary learned using K-SVD (INV-KSVD)

The original image used for this experiment is shown in Figure 5, and the reconstructed images, obtained with the three variants of the experiments, are shown in Figure 6.



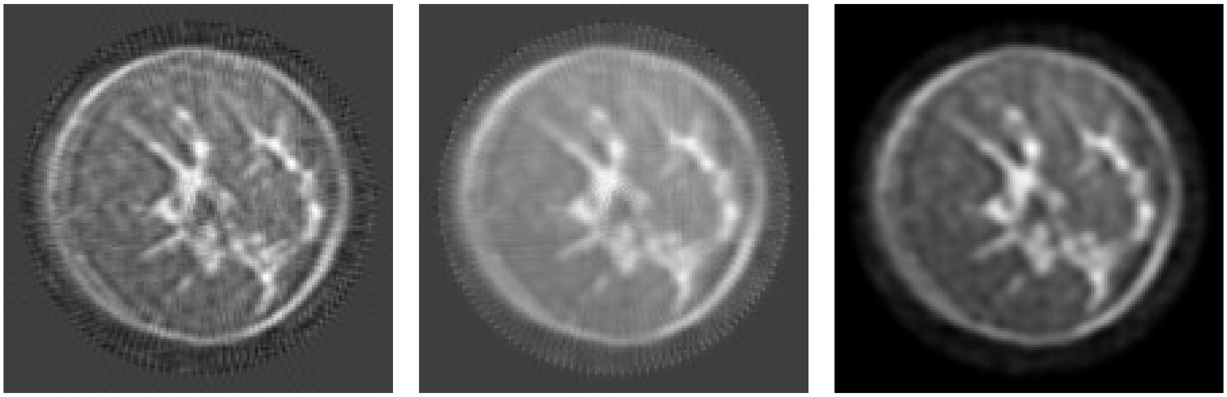
(a) INV (14.70 dB) (b) INV-DCT (16.37 dB) (c) INV-KSVD (16.47 dB)

Figure 6. Result of the inversion process for the three variants of the experiment, with 256 emitters/receivers and 256 atoms of 8×8 pixels for the dictionaries. Numbers in parentheses are the PSNRs of each image compared to the original one.

We see that the dictionary does not improve very much the results in this setup. However, one has to notice that the measurements give more than enough information to recover the image without the help of a dictionary. Indeed, for an image of 144×144 pixels, we have $256 * 255$ measurements, which is more than enough. However, note that the measurements we used were generated by our software, and thus noise-free. Running the same experiment in a real-world setup should better emphasize the benefits of dictionaries, as the structural information they provide should help getting rid of the noisy parts of the signal, thanks to the sparsity constraint.

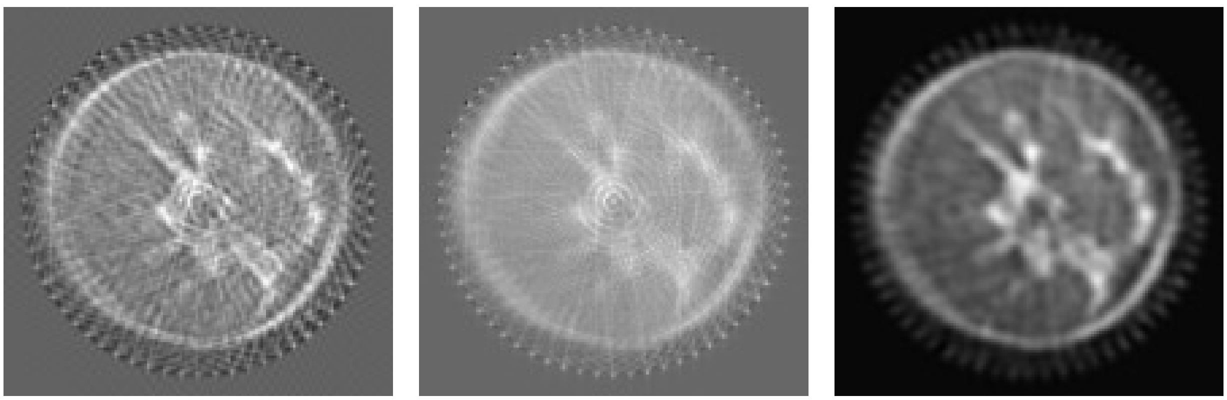
The primary goal of this project being to investigate whether dictionaries can help improve an image in poor measuring conditions, we tried the same experiment, but this time with less emitters/receivers. Instead of 256, we tried with 128, and then 64. Resulting images can be seen in Figures 5.2 and 5.2 respectively.

We see that this time the original version (INV) shows severe artifacts and a very poor quality, whereas the structure enforced by the dictionaries helps obtaining much nicer results. In fact, the difference between 128 and 256 emitters/receiver is barely noticeable with the K-SVD dictionary. The difference between 128 and 64 emitters/receivers is even more dramatic: the INV version has heavy ray artifacts, where K-SVD shows a pretty smooth reconstruction.



(a) INV (13.08 dB) (b) INV-DCT (13.00 dB) (c) INV-KSVD (16.20 dB)

Figure 7. Result of the inversion process for the three variants of the experiment, with 128 emitters/receivers and 256 atoms of 8×8 pixels for the dictionaries. Numbers in parentheses are the PSNRs of each image compared to the original one.



(a) INV (9.99 dB) (b) INV-DCT (8.90 dB) (c) INV-KSVD (16.01 dB)

Figure 8. Result of the inversion process for the three variants of the experiment, with 64 emitters/receivers and 256 atoms of 8×8 pixels for the dictionaries. Numbers in parentheses are the PSNRs of each image compared to the original one.

Note that the results obtained with the DCT dictionary are not as good as the one obtained with the learned dictionary. This comes from the fact that for comparison purposes, we showed results with the same parameters (patch size and number of atoms) for both DCT and K-SVD dictionaries, though these were not the optimal ones for DCT. Detailed results presented below indicated that better results can be achieved with a DCT dictionary.

In any case, more detailed experiments should be performed to determine the exact influence of the learning algorithm on the results of the inversion problem. The effects of the dictionary parameters (number of atoms, patch size) should also be further investigated.

6. CONCLUSION

In this article, we introduced the idea of using dictionaries in the context of a tomography setup. We first described the problem of sparse coding and its different solutions in Section 2. Then, we presented in Section 3 the two approaches for choosing a dictionary, either by picking a set of predefined functions, or by crafting it using some problem-specific data. We listed the different techniques for learning a dictionary, and illustrated our choice of one methods of each kind, namely the Discrete Cosine Transform as a complete basis, and K-SVD as learning algorithm. We introduced our tomography setup in Section 4, first in its original form, and then modified to make use of a dictionary. We also explained how our dictionaries defined for local atoms are generalized to be applied on whole images. Finally, we presented some interesting results in Section 5, in which dictionaries were able to help getting rid of downsampling and ray artifacts when the number of measurements was reduced. This shows promising applications for increasing the reconstruction resolution and quality when the number of measurements is limited. Some results with different combinations of number of measurements, number of atoms and patch sizes were also exhibited, but a formal and systematic study of the influence of these parameters still remains to be done.

6.1 Future work

As said above, the exploration of the influence of the dictionary structure (complete basis or learning-based? Which learning algorithm?), as well as the dictionary parameters (number of atoms, patch size) should be further investigated. More specifically, we would like to try dictionaries learned using Maximum Likelihood, as well as the 2D convolutional sparsenet developed by Culpepper.[11]

Moreover, the tomography setup on which we ran our reconstruction experiments was a very simplistic one: straight rays, no noise, etc. Further experiments should be run using a more complex setup, for instance with bent rays, to explore the effect of dictionaries in these cases.

REFERENCES

[1] Tropp, J. A., "Greed is good: Algorithmic results for sparse approximation," *IEEE Transactions on Information Theory* **50**, 2231–2242 (2004).

[2] Ahmed, N., Natarajan, T., and Rao, K., "Discrete cosine transform," *Computers, IEEE Transactions on* **C-23**, 90 –93 (jan. 1974).

[3] Bottou, L., "Online algorithms and stochastic approximations," in [*Online Learning and Neural Networks*], Saad, D., ed., Cambridge University Press, Cambridge, UK (1998).

[4] Mairal, J., Bach, F., Ponce, J., and Sapiro, G., "Online learning for matrix factorization and sparse coding," *Journal of Machine Learning Research* **11**, 19–60 (January 2010).

[5] Zou, H., Hastie, T., and Tibshirani, R., "Sparse principal component analysis," *Journal of Computational and Graphical Statistics* **15**, 2006 (2004).

[6] Lewicki, M. S. and Sejnowski, T. J., "Learning overcomplete representations," (2000).

[7] Elad, M. and Bruckstein, A. M., "A generalized uncertainty principle and sparse representation in pairs of bases," *IEEE Transactions on Information Theory* **48**, 2558–2567 (2002).

[8] Gribonval, R. and Nielsen, M., "Sparse Representations in Unions of Bases," Research Report RR-4642, INRIA (2002).

[9] Bruckstein, A. M., Donoho, D. L., and Elad, M., "From sparse solutions of systems of equations to sparse modeling of signals and images," *SIAM Review* **51**, 34–81 (February 2009).

[10] Rubinstein, R., Zibulevsky, M., and Elad, M., "Efficient implementation of the k-svd algorithm using batch orthogonal matching pursuit," tech. rep., Israel Institute of Technology (2008).

[11] Culpepper, B. J., *2D Convolutional Sparsenet* (2011 (Accessed September 2nd, 2011)). `https://github.com/jackculpepper/sparsenet-conv-2d`.